In [109…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score
```

In [110…
```
data = pd.read_csv(r'C:\Users\kriti\OneDrive\Desktop\Project\Heart Failure.csv')
```

In [111…
```
type (data)
```

Out[111]:
```
pandas.core.frame.DataFrame
```

In [112…
```
data.shape
```

Out[112]:
```
(299, 13)
```

In [113…
```
data.head()
```

Out[113]:

| | age | anaemia | creatinine_phosphokinase | diabetes | ejection_fraction | high_blood_pressure | platelet |
|---|---|---|---|---|---|---|---|
| **0** | 75.0 | 0 | 582 | 0 | 20 | 1 | 265000.0 |
| **1** | 55.0 | 0 | 7861 | 0 | 38 | 0 | 263358.0 |
| **2** | 65.0 | 0 | 146 | 0 | 20 | 0 | 162000.0 |
| **3** | 50.0 | 1 | 111 | 0 | 20 | 0 | 210000.0 |
| **4** | 65.0 | 1 | 160 | 1 | 20 | 0 | 327000.0 |

In [114…
```
data.columns
```

Out[114]:
```
Index(['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
       'DEATH_EVENT'],
      dtype='object')
```

In [115…
```
categorical_variables = data[["anaemia", "diabetes", "high_blood_pressure", "sex", "sm
continuous_variables = data [["age", "creatinine_phosphokinase", "ejection_fraction",
# sex=> # 0 = female
        # 1 = male
```

In [116…
```
data.isna().sum()
```

Out[116]:
```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

In [117…
```
data.isnull().sum()
```

Out[117]:
```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

In [118…
```
continuous_variables.describe()
```

Out[118]:

| | age | creatinine_phosphokinase | ejection_fraction | platelets | serum_creatinine | serum |
|---|---|---|---|---|---|---|
| **count** | 299.000000 | 299.000000 | 299.000000 | 299.000000 | 299.00000 | 29 |
| **mean** | 60.833893 | 581.839465 | 38.083612 | 263358.029264 | 1.39388 | 13 |
| **std** | 11.894809 | 970.287881 | 11.834841 | 97804.236869 | 1.03451 | |
| **min** | 40.000000 | 23.000000 | 14.000000 | 25100.000000 | 0.50000 | 11 |
| **25%** | 51.000000 | 116.500000 | 30.000000 | 212500.000000 | 0.90000 | 13 |
| **50%** | 60.000000 | 250.000000 | 38.000000 | 262000.000000 | 1.10000 | 13 |
| **75%** | 70.000000 | 582.000000 | 45.000000 | 303500.000000 | 1.40000 | 14 |
| **max** | 95.000000 | 7861.000000 | 80.000000 | 850000.000000 | 9.40000 | 14 |

In [119…
```
categorical_variables.describe()
```

Out[119]:

| | anaemia | diabetes | high_blood_pressure | sex | smoking |
|---|---|---|---|---|---|
| **count** | 299.000000 | 299.000000 | 299.000000 | 299.000000 | 299.00000 |
| **mean** | 0.431438 | 0.418060 | 0.351171 | 0.648829 | 0.32107 |
| **std** | 0.496107 | 0.494067 | 0.478136 | 0.478136 | 0.46767 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **50%** | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.00000 |
| **75%** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.00000 |
| **max** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.00000 |

In [120…

```
data.groupby("DEATH_EVENT").count()
```

Out[120]:

| DEATH_EVENT | age | anaemia | creatinine_phosphokinase | diabetes | ejection_fraction | high_blood_pressu |
|---|---|---|---|---|---|---|
| **0** | 203 | 203 | 203 | 203 | 203 | 2 |
| **1** | 96 | 96 | 96 | 96 | 96 | |

In [121…

```
age = data [["age"]]
platelets = data [["platelets"]]
```

In [122…

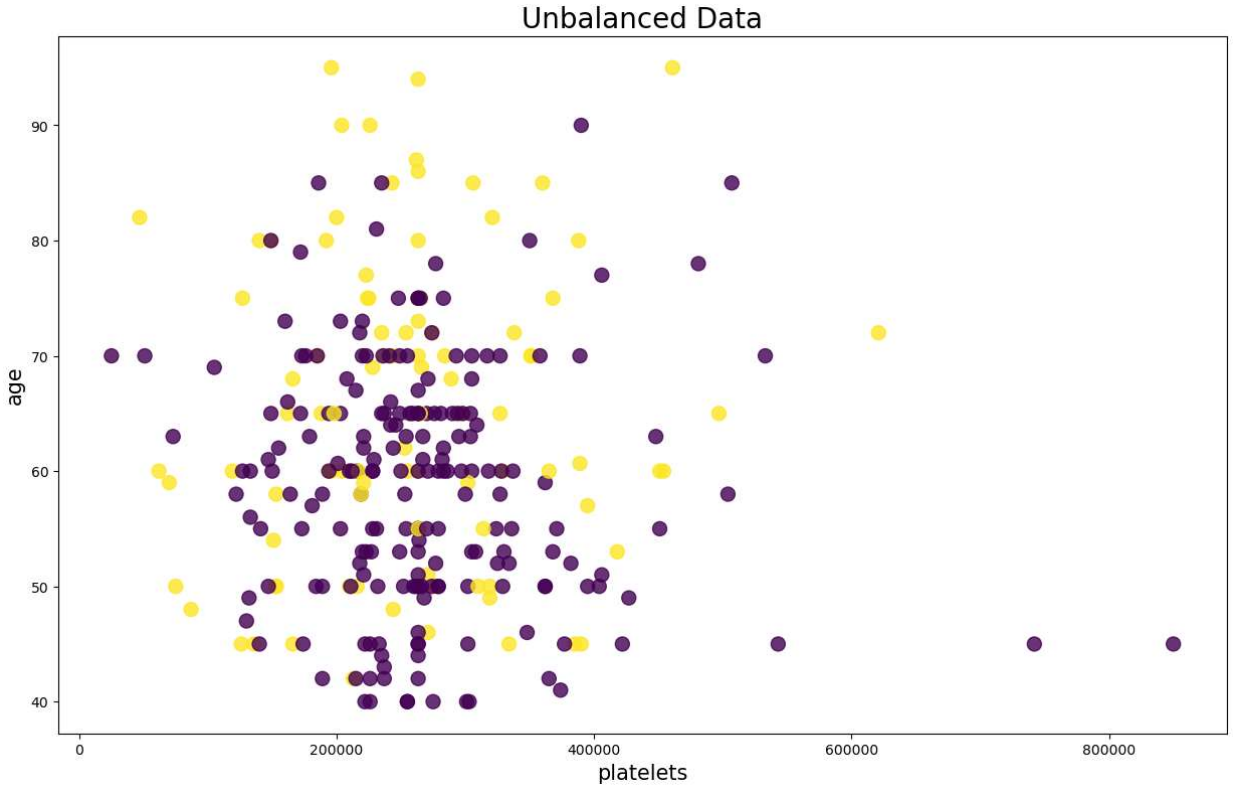
```
type(data['age'])
```

Out[122]:

```
pandas.core.series.Series
```

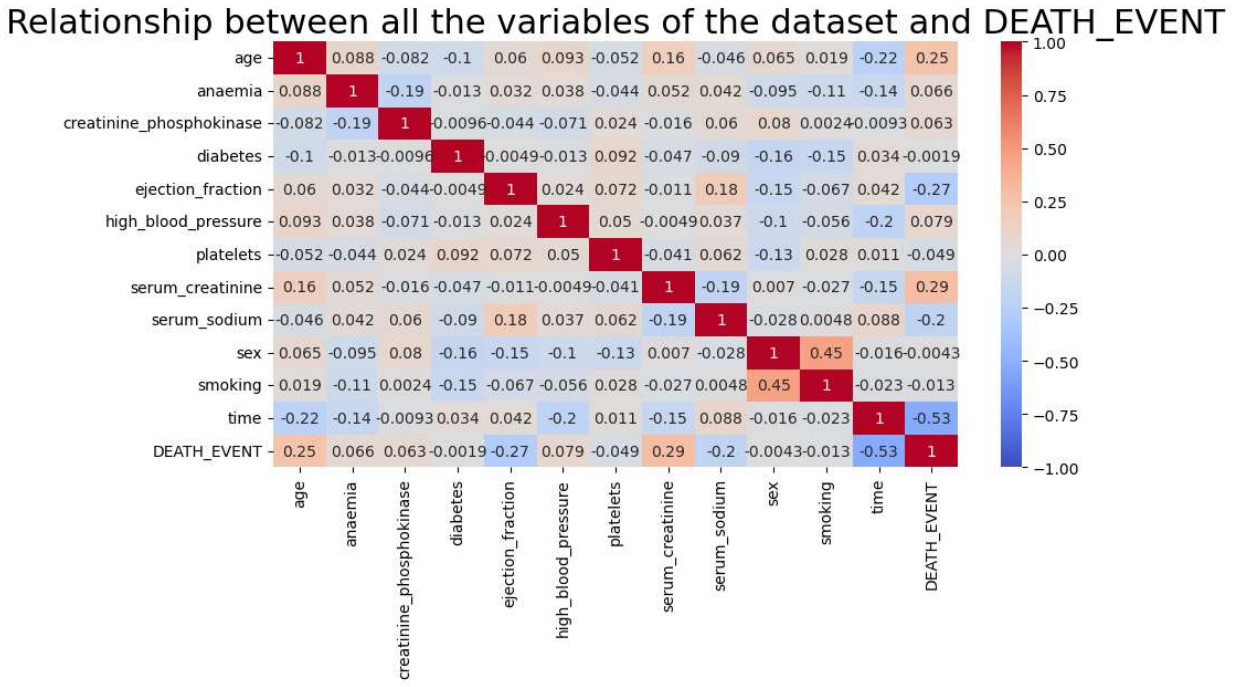
In [123…

```
plt.figure(figsize=(15,9))
plt.scatter(platelets, age, c = data["DEATH_EVENT"], s=102, alpha=0.8)
plt.xlabel("platelets", fontsize=15, color= "black")
plt.ylabel("age", fontsize= 15, color= "black")
plt.title("Unbalanced Data", fontsize=20)
plt.show()
```




In [124…

```
plt.figure(figsize=(10,5))
sns.heatmap(data.corr(), vmin=-1, vmax=1, cmap="coolwarm", annot=True)
plt.title("Relationship between all the variables of the dataset and DEATH_EVENT", for
plt.show()
```




In [125…

```
data.corr()
```

Out[125]:

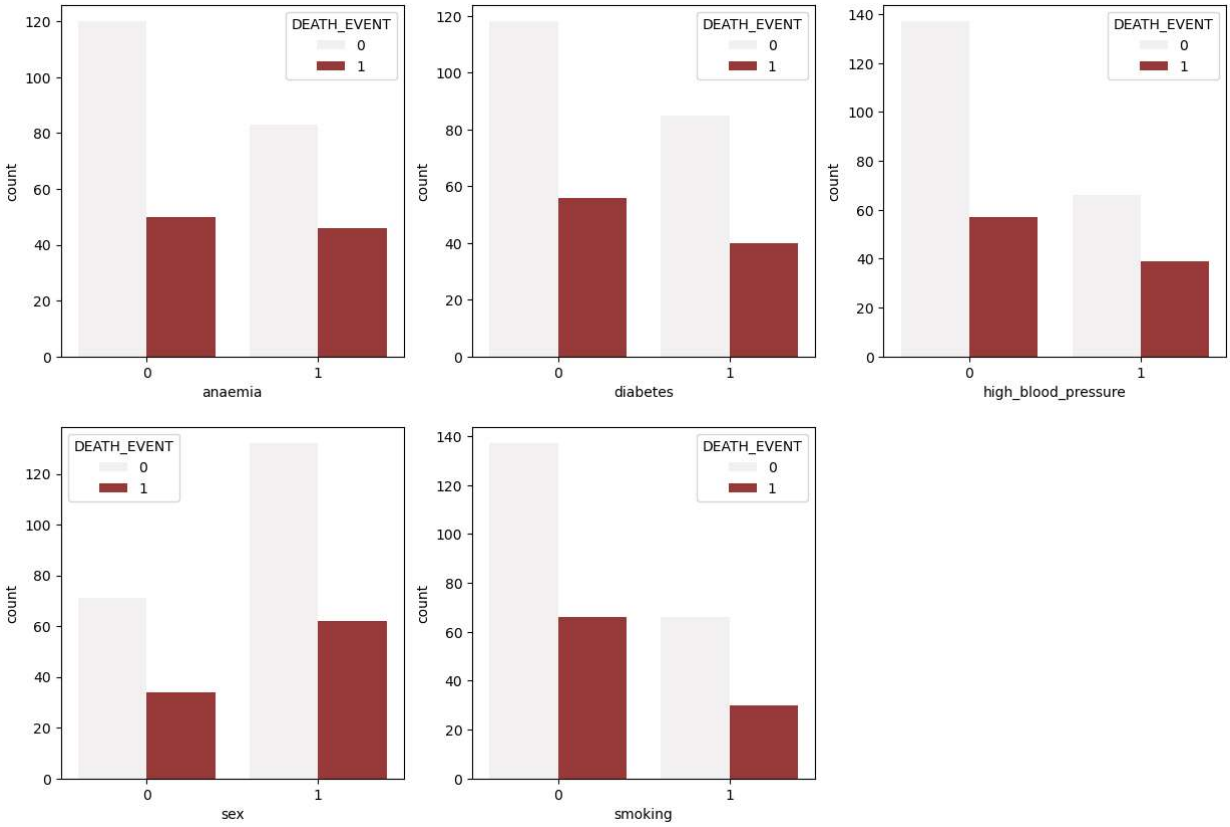
| | age | anaemia | creatinine_phosphokinase | diabetes | ejection_fraction |
|---|---|---|---|---|---|
| **age** | 1.000000 | 0.088006 | -0.081584 | -0.101012 | 0.060098 |
| **anaemia** | 0.088006 | 1.000000 | -0.190741 | -0.012729 | 0.031557 |
| **creatinine_phosphokinase** | -0.081584 | -0.190741 | 1.000000 | -0.009639 | -0.044080 |
| **diabetes** | -0.101012 | -0.012729 | -0.009639 | 1.000000 | -0.004850 |
| **ejection_fraction** | 0.060098 | 0.031557 | -0.044080 | -0.004850 | 1.000000 |
| **high_blood_pressure** | 0.093289 | 0.038182 | -0.070590 | -0.012732 | 0.024445 |
| **platelets** | -0.052354 | -0.043786 | 0.024463 | 0.092193 | 0.072177 |
| **serum_creatinine** | 0.159187 | 0.052174 | -0.016408 | -0.046975 | -0.011302 |
| **serum_sodium** | -0.045966 | 0.041882 | 0.059550 | -0.089551 | 0.175902 |
| **sex** | 0.065430 | -0.094769 | 0.079791 | -0.157730 | -0.148386 |
| **smoking** | 0.018668 | -0.107290 | 0.002421 | -0.147173 | -0.067315 |
| **time** | -0.224068 | -0.141414 | -0.009346 | 0.033726 | 0.041729 |
| **DEATH_EVENT** | 0.253729 | 0.066270 | 0.062728 | -0.001943 | -0.268603 |

In [126…

```
categorical_data = data[["anaemia", "diabetes", "high_blood_pressure", "sex", "smoking
continuous_data = data [["age", "creatinine_phosphokinase", "ejection_fraction", "plat
```

In [127…

```
plt.figure(figsize=(15,10))
for i, cat in enumerate (categorical_data):
    plt.subplot(2,3,i+1)
    sns.countplot(data= data, x=cat, hue= "DEATH_EVENT" , color="brown")
plt.show()
```



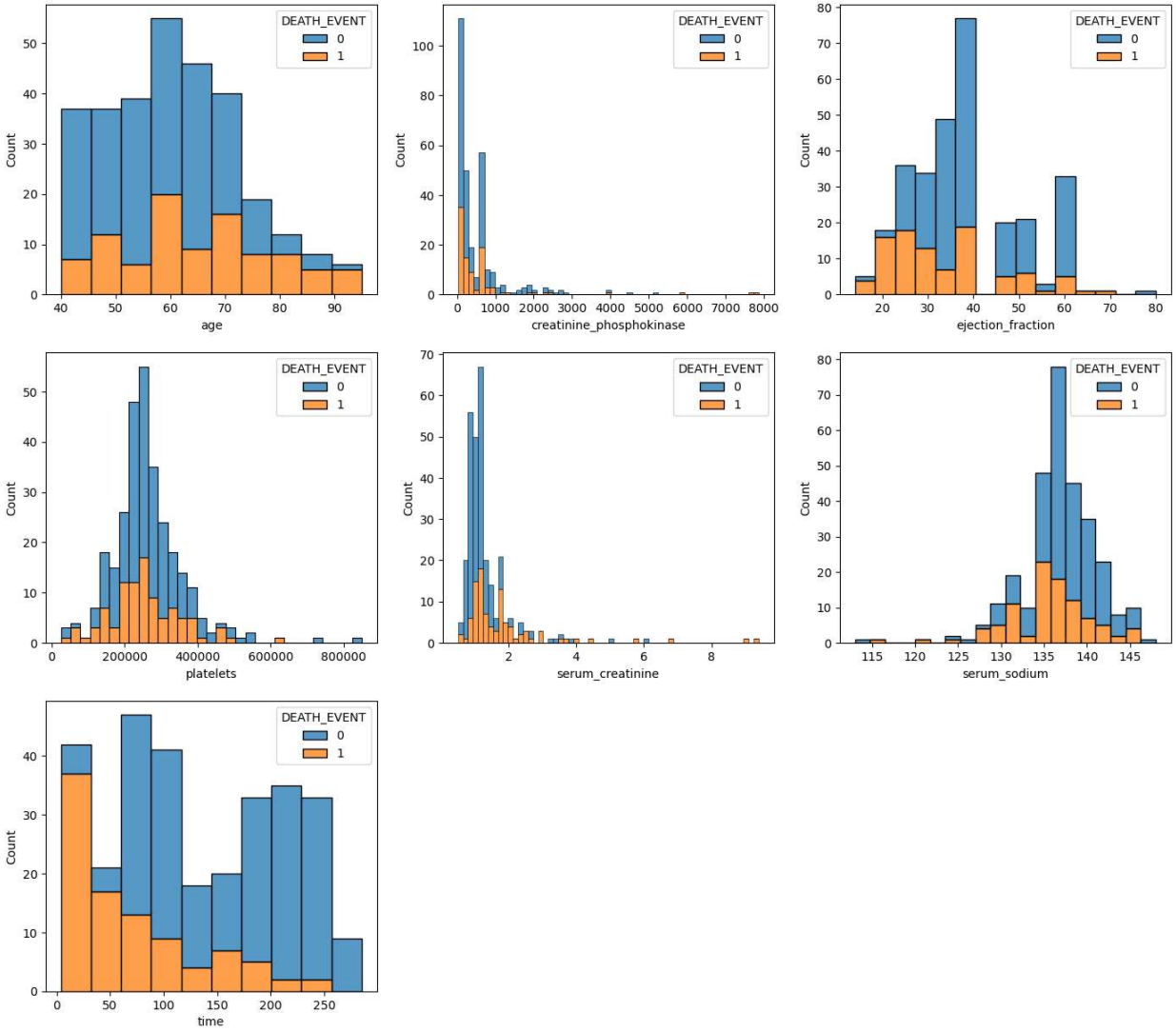

In [128…

```
plt.figure(figsize=(20,15))
plt.subplot(2,2,1)
sns.countplot(data= data, x='anaemia', hue= "DEATH_EVENT")
plt.subplot(2,2,4)
sns.countplot(data = data, x='diabetes', hue "DEATH_EVENT")
```

```
  Cell In[128], line 5
    sns.countplot(data = data, x='diabetes', hue "DEATH_EVENT")
                                                 ^
SyntaxError: positional argument follows keyword argument
```

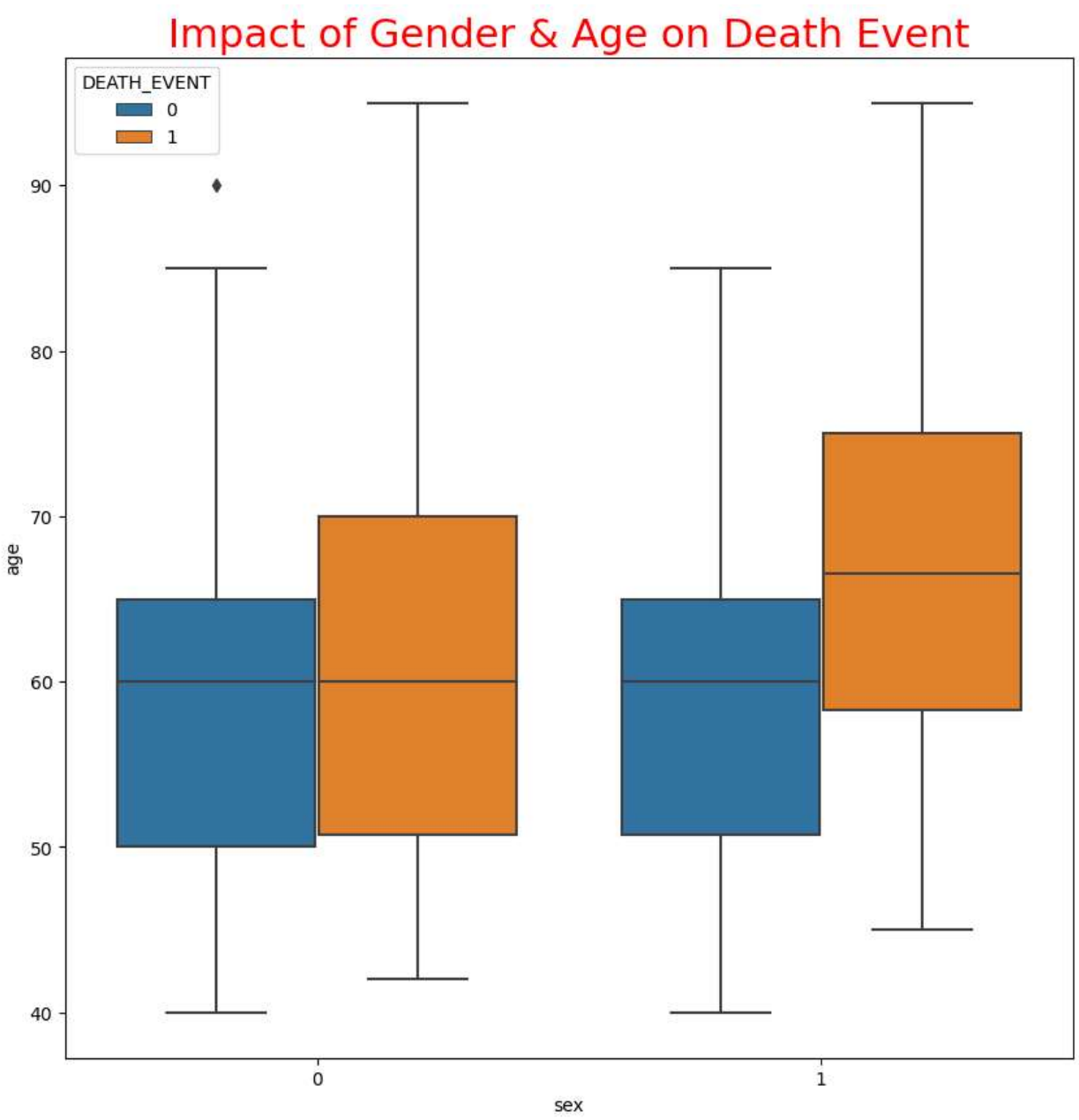
In [129…

```
# Plotting the impact of continuous variables on DEATH_EVENT

plt.figure(figsize=(17,15))
for j,con in enumerate(continuous_data):
    plt.subplot(3,3,j+1)
    sns.histplot(data = data, x= con, hue = "DEATH_EVENT", multiple="stack")
plt.show()
```



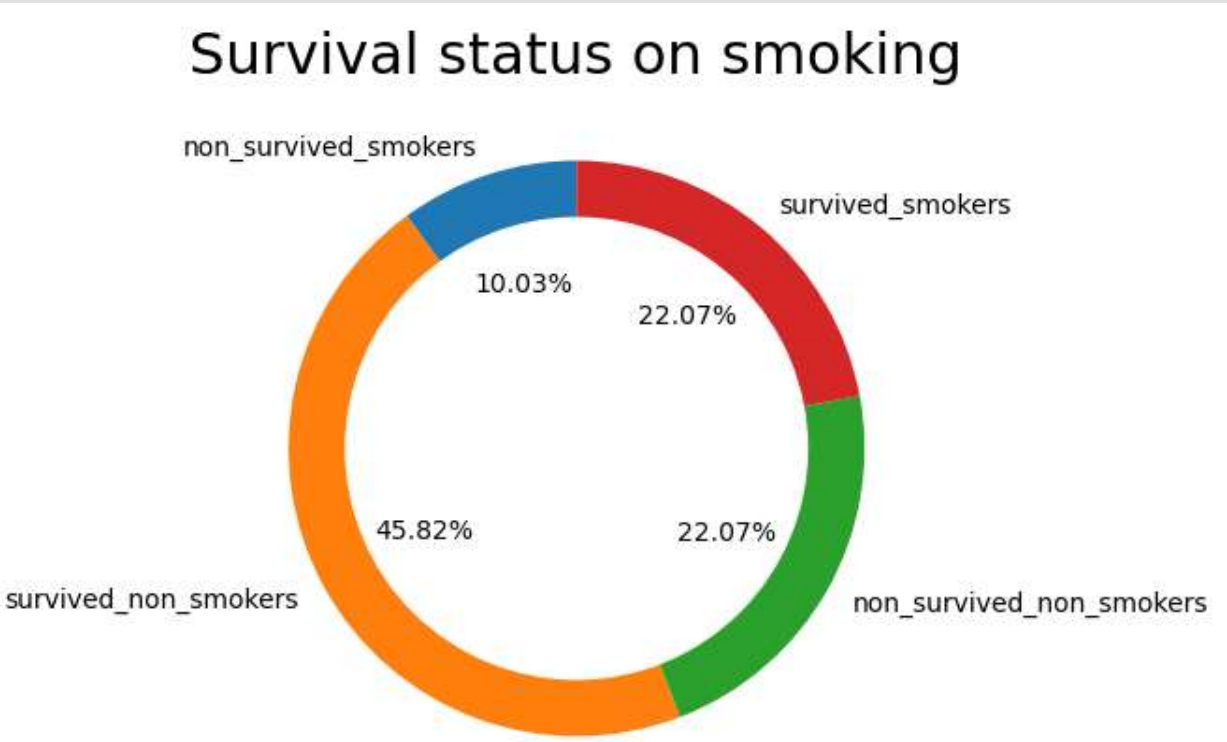

In [130…

```
plt.figure(figsize=(10,10))
sns.boxplot(data=data, x="sex", y="age", hue= "DEATH_EVENT")
plt.title("Impact of Gender & Age on Death Event", fontsize=22, color="red")
plt.show()
```




In [131…

```
# Analyzing the survival status on smoking

smokers = data[data["smoking"]==1]
non_smokers = data[data["smoking"]==0]

non_survived_smokers = smokers[smokers["DEATH_EVENT"]==1]
survived_non_smokers = non_smokers[non_smokers["DEATH_EVENT"]==0]
non_survived_non_smokers = non_smokers[non_smokers["DEATH_EVENT"]==1]
survived_smokers = smokers[smokers["DEATH_EVENT"]==0]

smoking_data = [len(non_survived_smokers), len(survived_non_smokers),len(non_survived_
smoking_labels = ["non_survived_smokers", "survived_non_smokers", "non_survived_non_sm

plt.figure(figsize=(5,5))
plt.pie(smoking_data, labels = smoking_labels, autopct='%.2f%%', startangle=90)
circle = plt.Circle((0,0), 0.8, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on smoking", fontsize=22)
plt.show()
```




In [132…
```
type(non_smokers)
```

Out[132]:
```
pandas.core.frame.DataFrame
```

In [133…
```
smokers[smokers["DEATH_EVENT"]==1]
```

Out[133]:

| | age | anaemia | creatinine_phosphokinase | diabetes | ejection_fraction | high_blood_pressure | plate |
|---|---|---|---|---|---|---|---|
| **2** | 65.0 | 0 | 146 | 0 | 20 | 0 | 162000 |
| **5** | 90.0 | 1 | 47 | 0 | 40 | 1 | 204000 |
| **7** | 60.0 | 1 | 315 | 1 | 60 | 0 | 454000 |
| **9** | 80.0 | 1 | 123 | 0 | 35 | 1 | 388000 |
| **10** | 75.0 | 1 | 81 | 0 | 38 | 1 | 368000 |
| **11** | 62.0 | 0 | 231 | 0 | 25 | 1 | 253000 |
| **22** | 68.0 | 1 | 220 | 0 | 35 | 1 | 289000 |
| **25** | 80.0 | 0 | 148 | 1 | 38 | 0 | 149000 |
| **27** | 70.0 | 0 | 122 | 1 | 45 | 1 | 284000 |
| **29** | 82.0 | 0 | 70 | 1 | 30 | 0 | 200000 |
| **40** | 70.0 | 0 | 582 | 0 | 20 | 1 | 263358 |
| **41** | 50.0 | 0 | 124 | 1 | 30 | 1 | 153000 |
| **42** | 70.0 | 0 | 571 | 1 | 45 | 1 | 185000 |
| **45** | 50.0 | 0 | 582 | 1 | 38 | 0 | 310000 |
| **47** | 60.0 | 0 | 582 | 1 | 38 | 1 | 451000 |
| **54** | 60.0 | 1 | 260 | 1 | 38 | 0 | 255000 |
| **58** | 49.0 | 0 | 789 | 0 | 20 | 1 | 319000 |
| **59** | 72.0 | 0 | 364 | 1 | 20 | 1 | 254000 |
| **65** | 60.0 | 0 | 68 | 0 | 20 | 0 | 119000 |
| **67** | 72.0 | 1 | 110 | 0 | 25 | 0 | 274000 |
| **72** | 85.0 | 0 | 5882 | 0 | 35 | 0 | 243000 |
| **74** | 69.0 | 0 | 582 | 0 | 20 | 0 | 266000 |
| **75** | 60.0 | 1 | 47 | 0 | 20 | 0 | 204000 |
| **105** | 72.0 | 1 | 328 | 0 | 30 | 1 | 621000 |
| **110** | 85.0 | 0 | 129 | 0 | 60 | 0 | 306000 |
| **144** | 72.0 | 1 | 943 | 0 | 25 | 1 | 338000 |
| **181** | 59.0 | 1 | 176 | 1 | 25 | 0 | 221000 |
| **182** | 65.0 | 0 | 395 | 1 | 25 | 0 | 265000 |
| **184** | 58.0 | 1 | 145 | 0 | 25 | 0 | 219000 |
| **266** | 55.0 | 0 | 1199 | 0 | 20 | 0 | 263358 |

In [134…

```
(len(non_survived_smokers)/299)*100
```

Out[134]:

```
10.033444816053512
```

In [135…
```
smoking_data
```

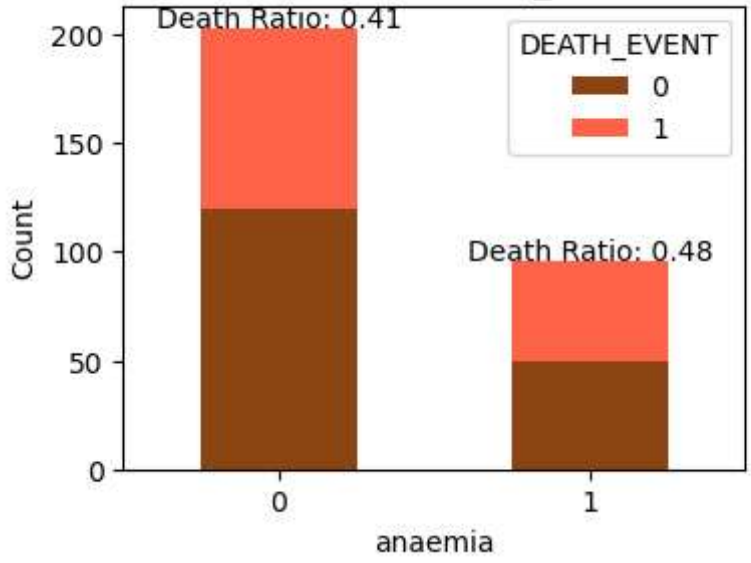
Out[135]:
```
[30, 137, 66, 66]
```

In [136…
```
smoking_labels
```

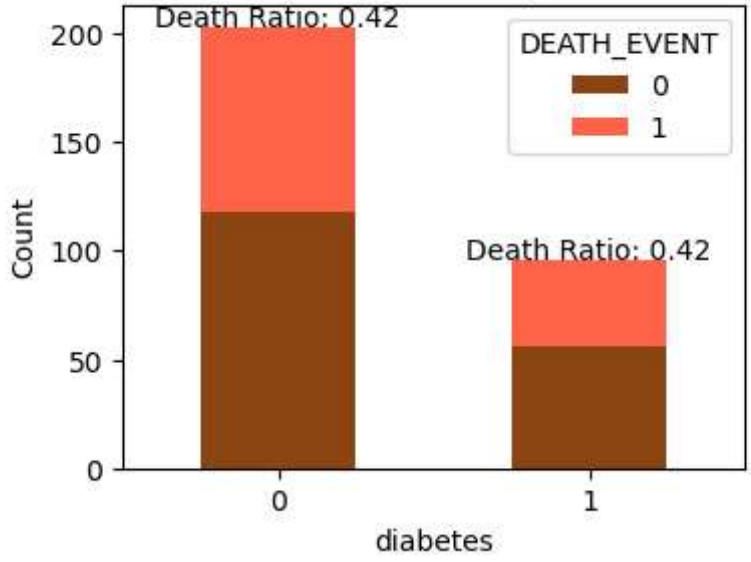
Out[136]:
```
['non_survived_smokers',
 'survived_non_smokers',
 'non_survived_non_smokers',
 'survived_smokers']
```

In [181…
```
import pandas as pd
import matplotlib.pyplot as plt

categorical_data = data[["anaemia", "diabetes", "high_blood_pressure", "sex", "smoking

# Grouping by each categorical variable and 'DEATH_EVENT' and counting occurrences
grouped = categorical_data.groupby(['DEATH_EVENT'])

# List of colors for bars
colors = ['#8B4513', '#FF6347']  # Brown and Red colors

# Plotting stacked bar charts for each categorical variable
for col in categorical_data.columns[:-1]:  # Exclude 'DEATH_EVENT'
    # Count occurrences of DEATH_EVENT in each category
    counts = grouped[col].value_counts().unstack().fillna(0)

    # Compute the death ratio for each category
    total = counts.sum(axis=1)
    death_ratio = counts.iloc[:, 1] / total

    # Plot stacked bar chart
    counts.plot(kind='bar', stacked=True, color=colors, figsize=(4, 3))
    plt.title(f'Stacked Column Chart of DEATH_EVENT for {col}')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.xticks(rotation=0)

    # Display death ratio on each bar
    for i, value in enumerate(death_ratio):
        plt.text(i, counts.iloc[i].sum(), f"Death Ratio: {value:.2f}", color='black',

    plt.legend(title='DEATH_EVENT', loc='upper right')
    plt.show()
```

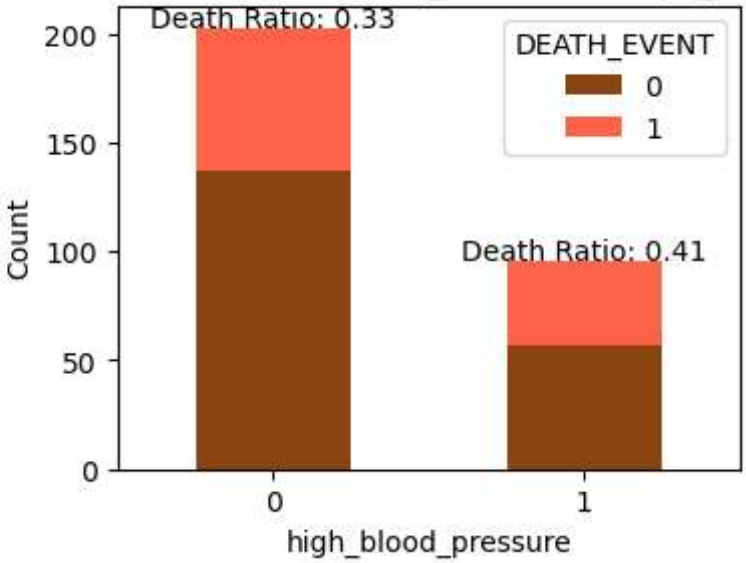
Stacked Column Chart of DEATH_EVENT for anaemia

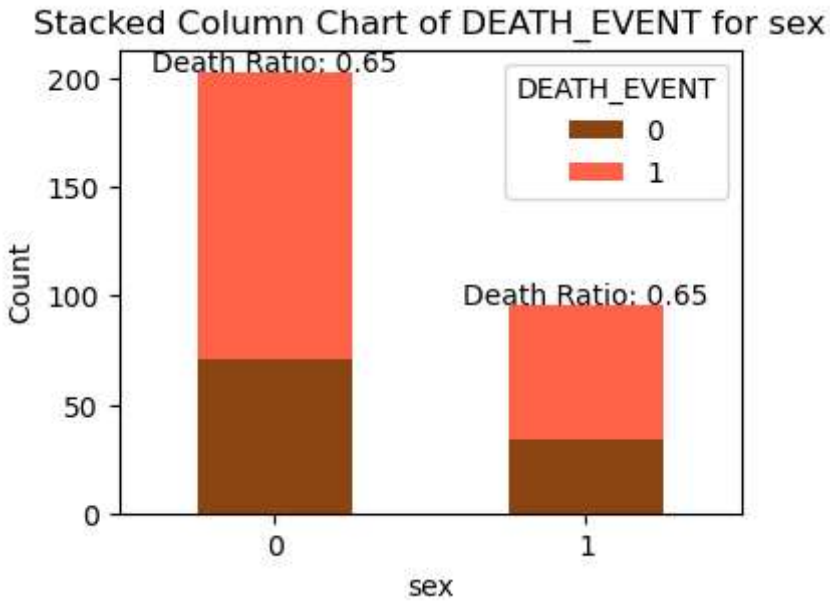


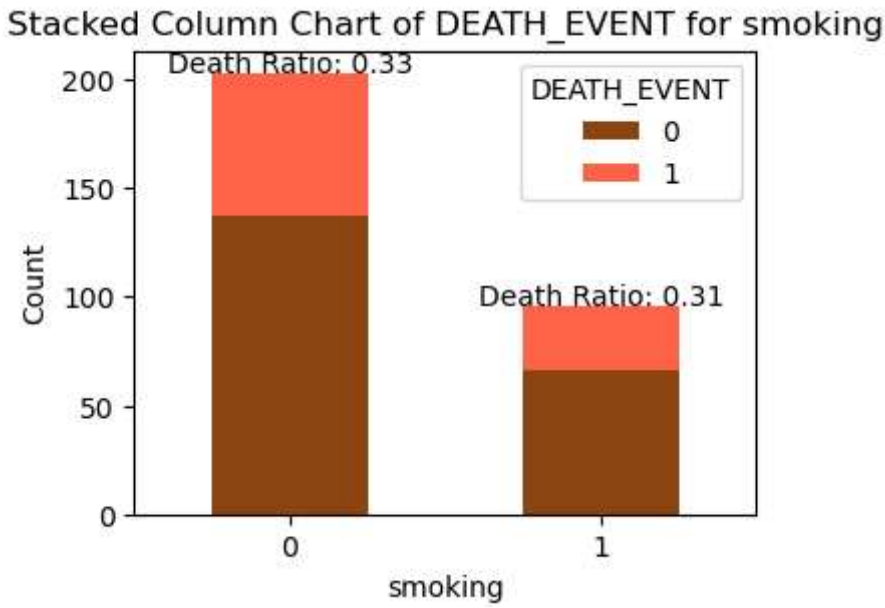

Stacked Column Chart of DEATH_EVENT for diabetes




Stacked Column Chart of DEATH_EVENT for high_blood_pressure

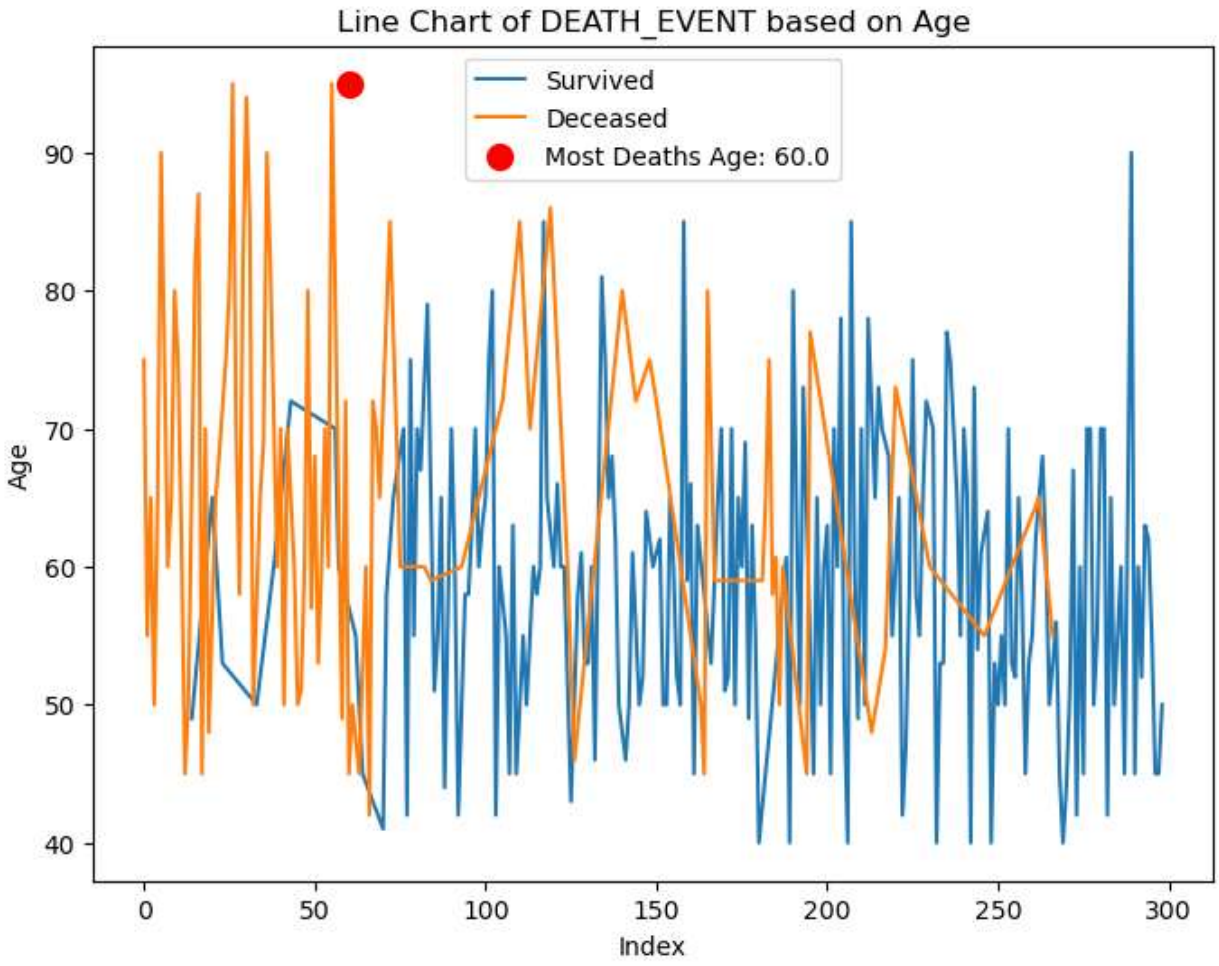




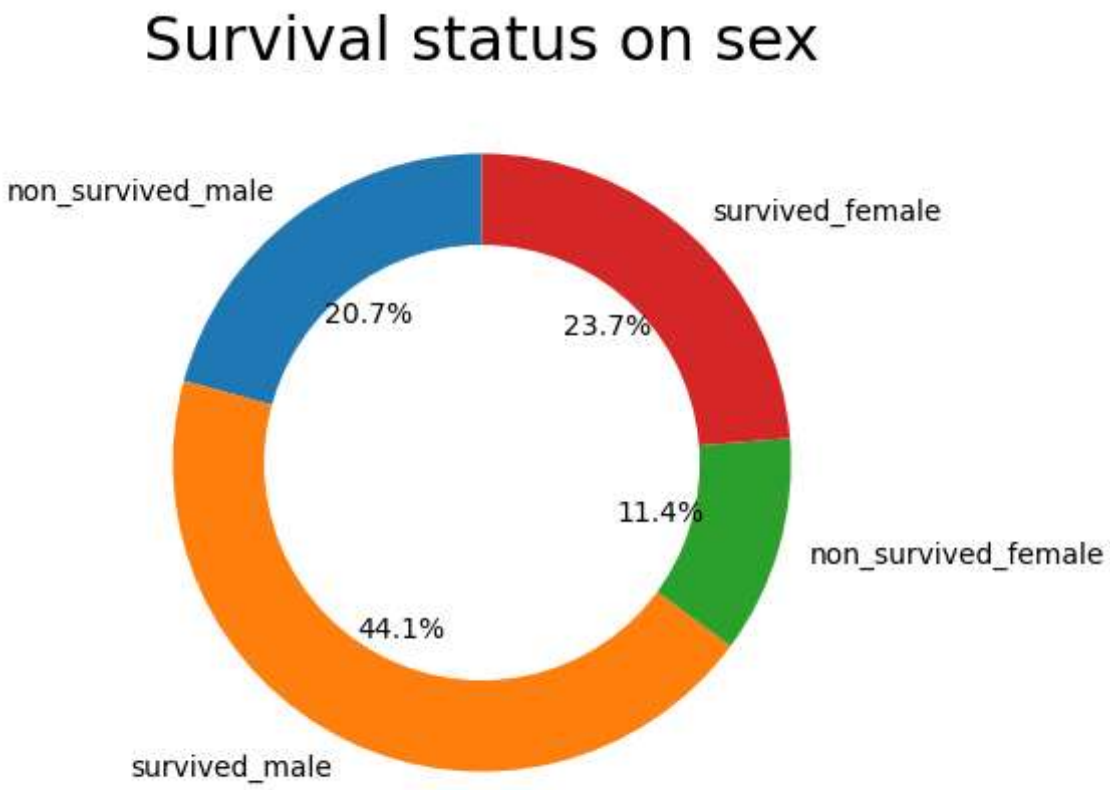

In [138…

```python
import matplotlib.pyplot as plt

deceased_age = data[data['DEATH_EVENT'] == 1]['age']  # Filter ages for deceased patie
most_deaths_age = deceased_age.value_counts().idxmax()  # Get age with most deaths

plt.figure(figsize=(8, 6))
plt.plot(data[data['DEATH_EVENT'] == 0]['age'], label='Survived')
plt.plot(data[data['DEATH_EVENT'] == 1]['age'], label='Deceased')
plt.scatter(most_deaths_age, max(deceased_age), color='red', label=f'Most Deaths Age:
plt.title('Line Chart of DEATH_EVENT based on Age')
plt.xlabel('Index')
plt.ylabel('Age')
plt.legend()
plt.show()
```



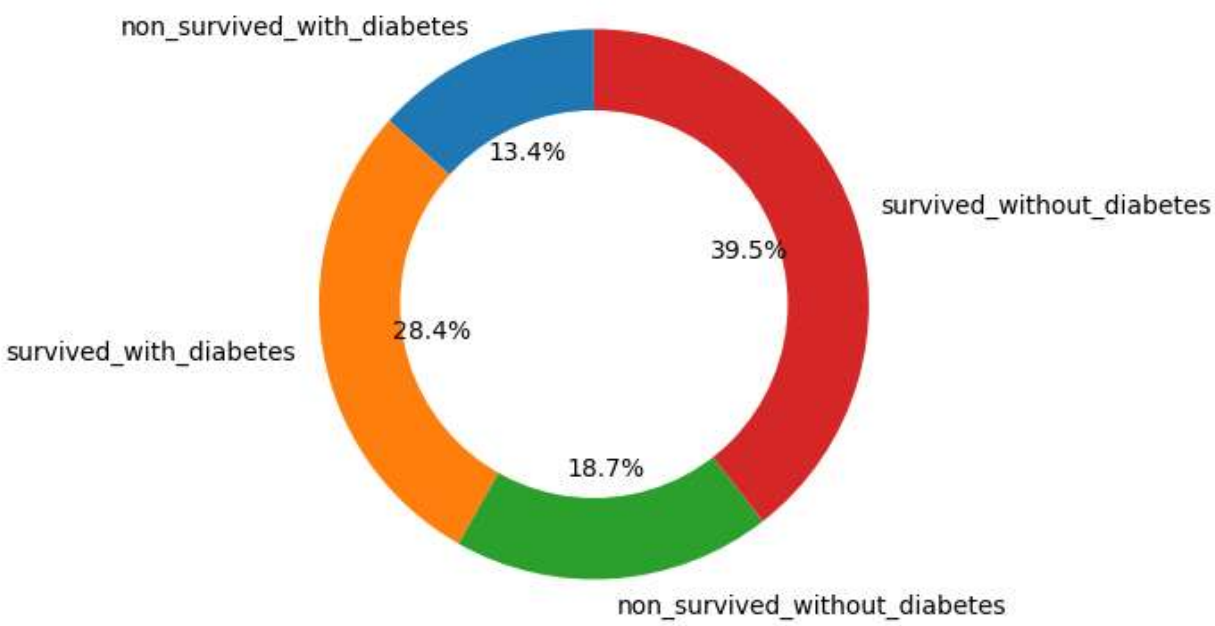

In [139…

```
# Analyzing the survival status on sex

male = data[data["sex"]==1]
female = data[data["sex"]==0]

non_survived_male = male[male["DEATH_EVENT"]==1]
survived_male = male[male["DEATH_EVENT"]==0]
non_survived_female = female[female["DEATH_EVENT"]==1]
survived_female = female[female["DEATH_EVENT"]==0]

sex_data = [len(non_survived_male), len(survived_male), len(non_survived_female),len(s
sex_labels = ["non_survived_male","survived_male","non_survived_female","survived_fema

plt.figure(figsize=(5,5))
plt.pie(sex_data, labels = sex_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on sex", fontsize=22)
plt.show()
```



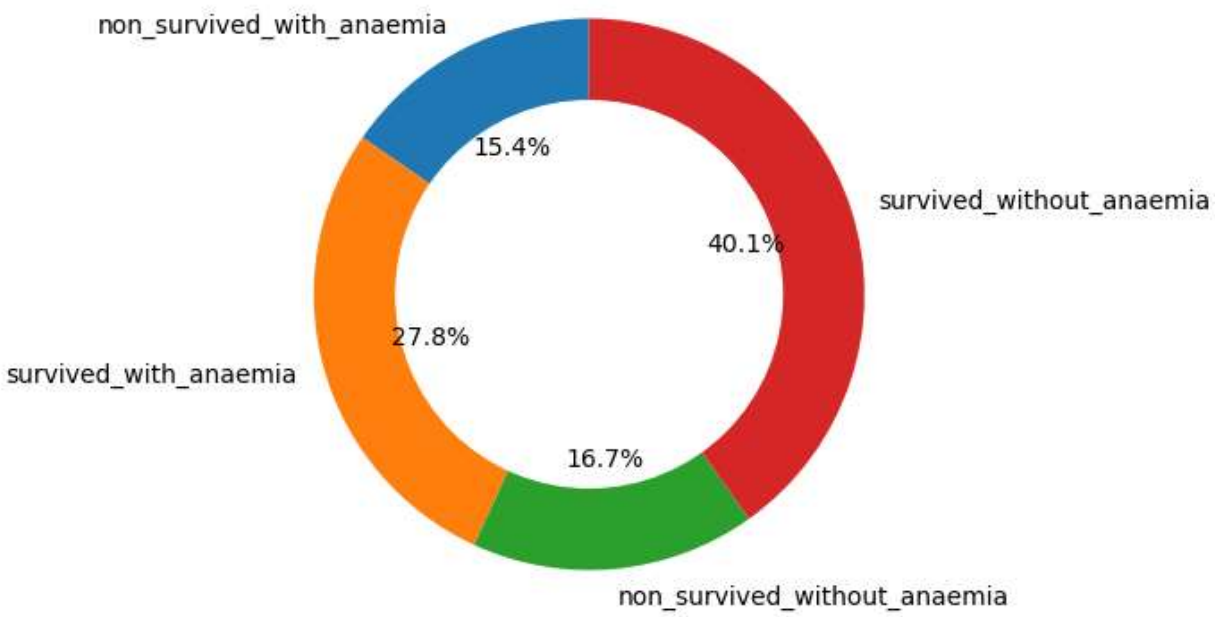

In [140…

```
# Analyzing the survival status on diabetes

with_diabetes = data[data["diabetes"]==1]
without_diabetes = data[data["diabetes"]==0]

non_survived_with_diabetes = with_diabetes[with_diabetes["DEATH_EVENT"]==1]
survived_with_diabetes = with_diabetes[with_diabetes["DEATH_EVENT"]==0]
non_survived_without_diabetes = without_diabetes[without_diabetes["DEATH_EVENT"]==1]
survived_without_diabetes = without_diabetes[without_diabetes["DEATH_EVENT"]==0]

diabetes_data = [len(non_survived_with_diabetes), len(survived_with_diabetes), len(nor
                 len(survived_without_diabetes)]
diabetes_labels = ["non_survived_with_diabetes","survived_with_diabetes","non_survived
                   "survived_without_diabetes"]

plt.figure(figsize=(5,5))
plt.pie(diabetes_data, labels = diabetes_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on diabetes", fontsize=22)
plt.show()
```



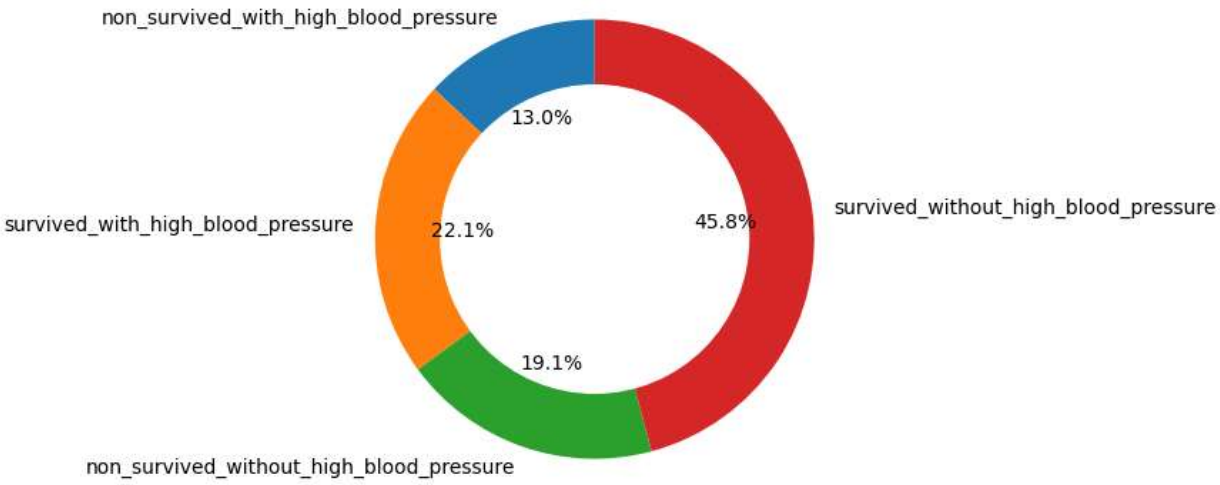

In [141…

```
# Analyzing the survival status on anaemia

with_anaemia = data[data["anaemia"]==1]
without_anaemia = data[data["anaemia"]==0]

non_survived_with_anaemia = with_anaemia[with_anaemia["DEATH_EVENT"]==1]
survived_with_anaemia = with_anaemia[with_anaemia["DEATH_EVENT"]==0]
non_survived_without_anaemia = without_anaemia[without_anaemia["DEATH_EVENT"]==1]
survived_without_anaemia = without_anaemia[without_anaemia["DEATH_EVENT"]==0]

anaemia_data = [len(non_survived_with_anaemia), len(survived_with_anaemia), len(non_su
                len(survived_without_anaemia)]
anaemia_labels = ["non_survived_with_anaemia","survived_with_anaemia","non_survived_wi
                  "survived_without_anaemia"]

plt.figure(figsize=(5,5))
plt.pie(anaemia_data, labels = anaemia_labels, autopct='%.1f%%', startangle=90)
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on anaemia", fontsize=22)
plt.show()
```




In [142…

```
# Analyzing the survival status on high blood pressure

with_high_blood_pressure = data[data["high_blood_pressure"]==1]
without_high_blood_pressure = data[data["high_blood_pressure"]==0]

non_survived_with_high_blood_pressure = with_high_blood_pressure[with_high_blood_press
survived_with_high_blood_pressure = with_high_blood_pressure[with_high_blood_pressure[
non_survived_without_high_blood_pressure = without_high_blood_pressure[without_high_bl
survived_without_high_blood_pressure = without_high_blood_pressure[without_high_blood_

high_blood_pressure_data = [len(non_survived_with_high_blood_pressure), len(survived_w
                            len(non_survived_without_high_blood_pressure), len(survive

high_blood_pressure_labels = ["non_survived_with_high_blood_pressure","survived_with_h
                  "non_survived_without_high_blood_pressure","survived_without_high_b

plt.figure(figsize=(5,5))
plt.pie(high_blood_pressure_data, labels = high_blood_pressure_labels, autopct='%.1f%%
circle = plt.Circle((0,0), 0.7, color="white")
p = plt.gcf()
p.gca().add_artist(circle)
plt.title("Survival status on high blood pressure", fontsize=22)
plt.show()
```




In [166…
```
x = data[["age","creatinine_phosphokinase","ejection_fraction","serum_creatinine","ser
y = data["DEATH_EVENT"]
```

In [167…
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=
```

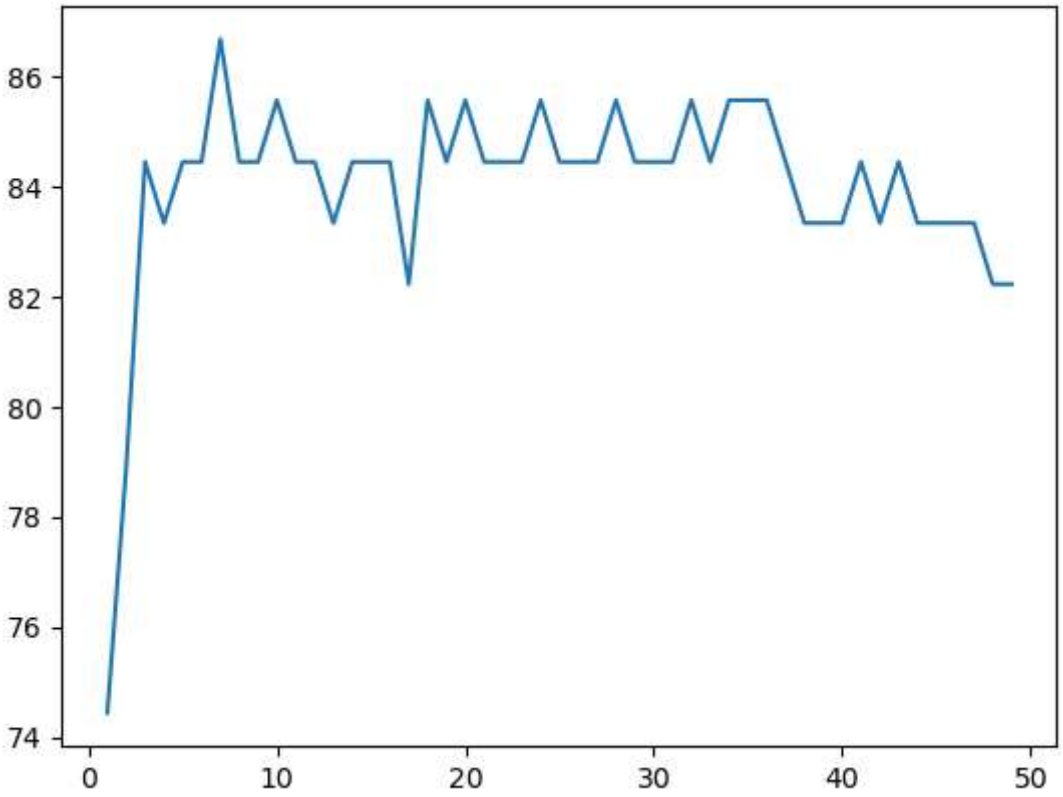
In [168…
```
# Data scaling
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

In [169…
```
 accuracy_list = [] # A list to save all the values from different models accuracy for
```

In [170…
```
lr_model = LogisticRegression()
lr_model.fit(x_train_scaled, y_train)
lr_prediction = lr_model.predict(x_test_scaled)
lr_accuracy = (round(accuracy_score(lr_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(lr_accuracy)
```

In [171…
```
# find the optimal value of k
knn_list = []
for k in range(1,50):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(x_train_scaled, y_train)
    knn_prediction = knn_model.predict(x_test_scaled)
    knn_accuracy = (round(accuracy_score(knn_prediction, y_test), 4) * 100)
    knn_list.append(knn_accuracy)
k = np.arange(1,50)
plt.plot(k, knn_list)
```

Out[171]:
```
[<matplotlib.lines.Line2D at 0x1dfa880f750>]
```




In [172…
```
knn_model = KNeighborsClassifier(n_neighbors=6)
knn_model.fit(x_train_scaled, y_train)
knn_prediction = knn_model.predict(x_test_scaled)
knn_accuracy = (round(accuracy_score(knn_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(knn_accuracy)
```

In [173…
```
dt_model = DecisionTreeClassifier(criterion="entropy", max_depth=2)
dt_model.fit(x_train_scaled, y_train)
dt_prediction = dt_model.predict(x_test_scaled)
dt_accuracy = (round(accuracy_score(dt_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(dt_accuracy)
```

In [174…
```
nb_model = GaussianNB()
nb_model.fit(x_train_scaled, y_train)
nb_prediction = nb_model.predict(x_test_scaled)
nb_accuracy = (round(accuracy_score(nb_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(nb_accuracy)
```

In [175…
```
rf_model = RandomForestClassifier()
rf_model.fit(x_train_scaled, y_train)
rf_prediction = rf_model.predict(x_test_scaled)
rf_accuracy = (round(accuracy_score(rf_prediction, y_test), 4) * 100) #percentage
accuracy_list.append(rf_accuracy)
```

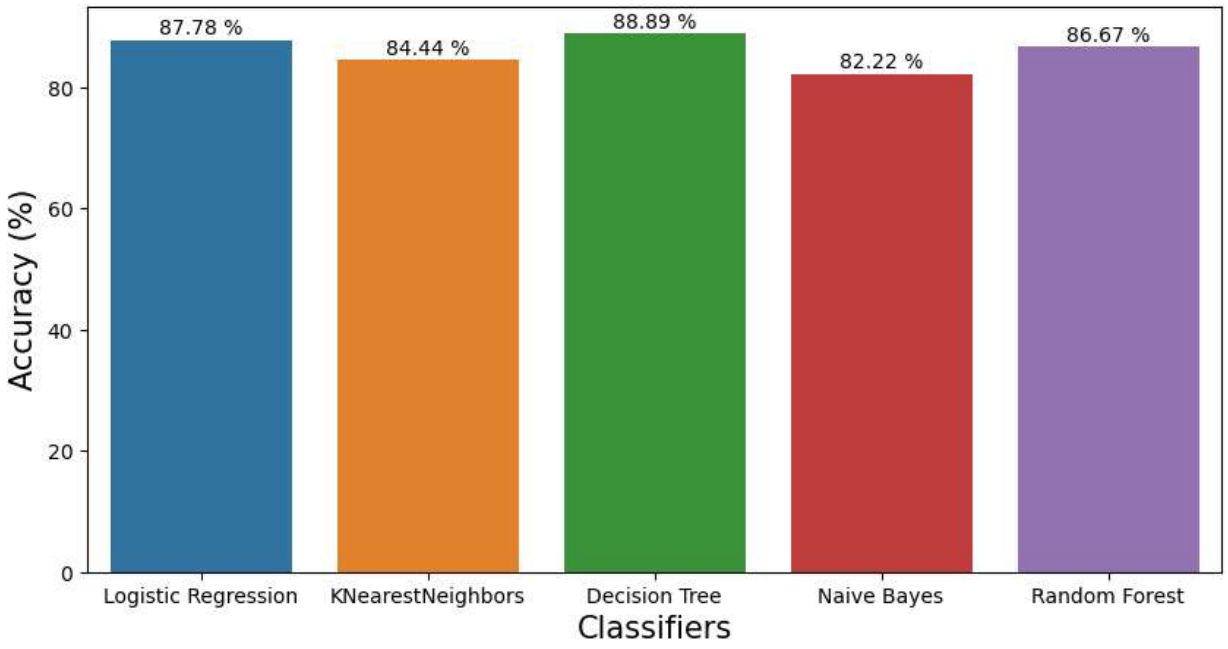
In [176…
```
accuracy_list
```

Out[176]:
```
[87.78, 84.44, 88.89, 82.22, 86.67]
```

In [177…
```
models = ["Logistic Regression","KNearestNeighbors","Decision Tree","Naive Bayes","Rar
```

In [178…

```
plt.figure(figsize=(10,5))
ax = sns.barplot(x=models, y=accuracy_list)
plt.xlabel("Classifiers", fontsize=15)
plt.ylabel("Accuracy (%)", fontsize=15)
for p in ax.patches:
    width = p.get_width()
    height = p.get_height()
    x = p.get_x()
    y = p.get_y()
    ax.annotate(f"{height} %", (x + width/2, y+ height*1.01), ha="center")
plt.show()
```




In [ ]: